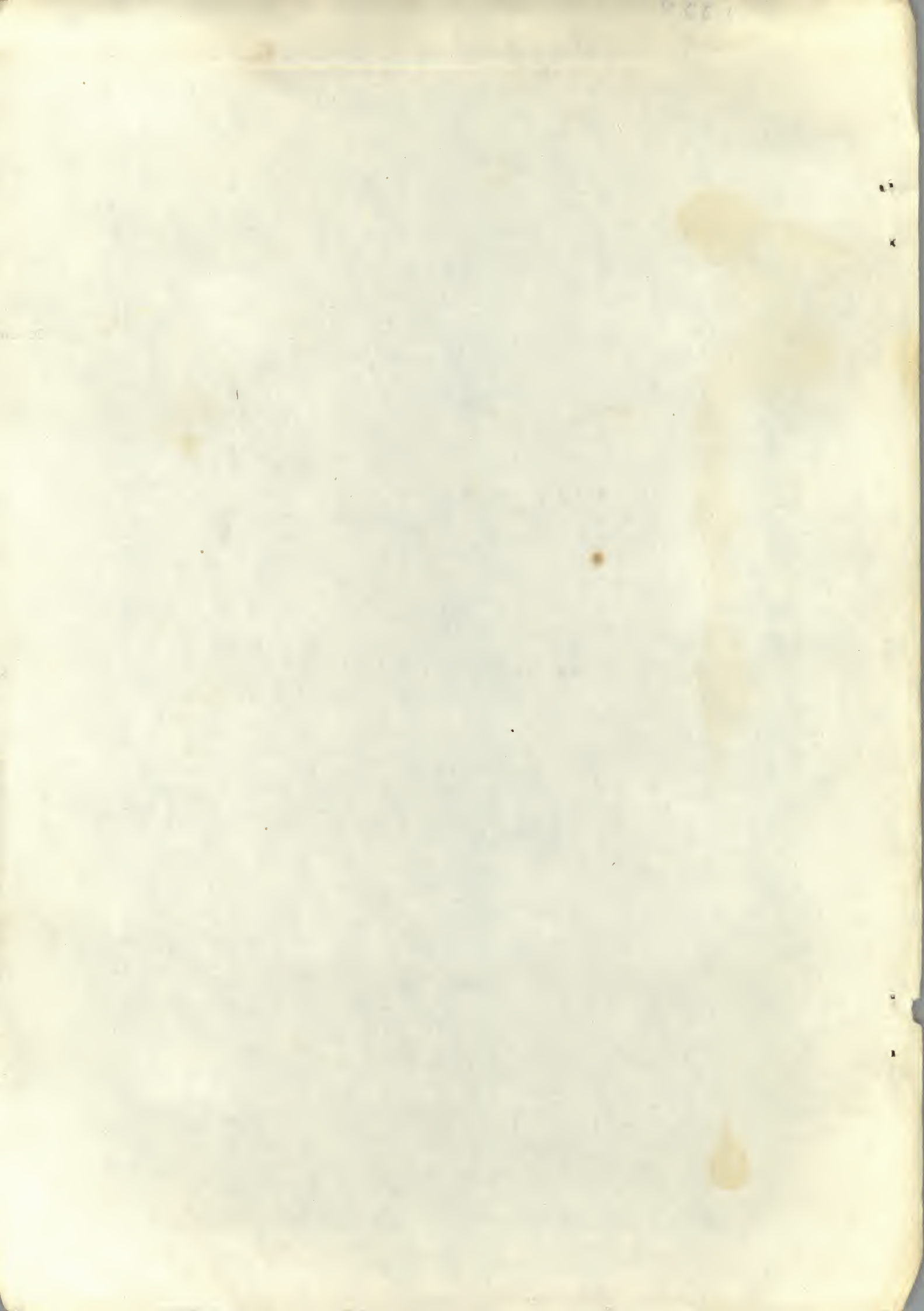


1 329

@
Kees de Groot
RC-LH

BOOK SEVEN

BASIC / RT



CONTENTS

	<u>Page</u>
INTRODUCTION	
CHAPTER 1 USING BASIC/RT	7-1
1.0 SCOPE COMMANDS	7-2
1.1 PLOT	7-2
1.2 DELAY	7-3
1.3 CLEAR	7-3
1.4 Display Buffer	7-4
1.5 USE	7-4
CHAPTER 2 CLOCK COMMANDS	7-6
2.0 GENERAL DESCRIPTION	7-6
2.1 SET RATE	7-6
2.2 SET CLOCK	7-7
2.3 TIM	7-7
2.4 WAITC	7-8
2.5 CLS and CLC	7-8
CHAPTER 3 A/D COMMANDS	7-10
3.1 ADC	7-10
3.2 REAL TIME	7-10
3.3 ACCEPT and REJECT	7-11
3.4 Buffer Capacity	7-11
3.5 ADB	7-12
CHAPTER 4 TEST AND PAUSE COMMANDS	7-14
4.1 TST	7-14
4.2 WAIT	7-14
CHAPTER 5 USER COMMANDS AND FUNCTIONS	7-15
5.1 User Command (UCOM)	7-15
5.2 User Coded Function	7-17
CHAPTER 6 ERROR MESSAGES	7-18
TABLES	
Table 7-1 Clock Enable Register Functions	7-9
7-2 BASIC/RT Error Messages	7-18

CHAPTER 1

USING BASIC/RT

INTRODUCTION

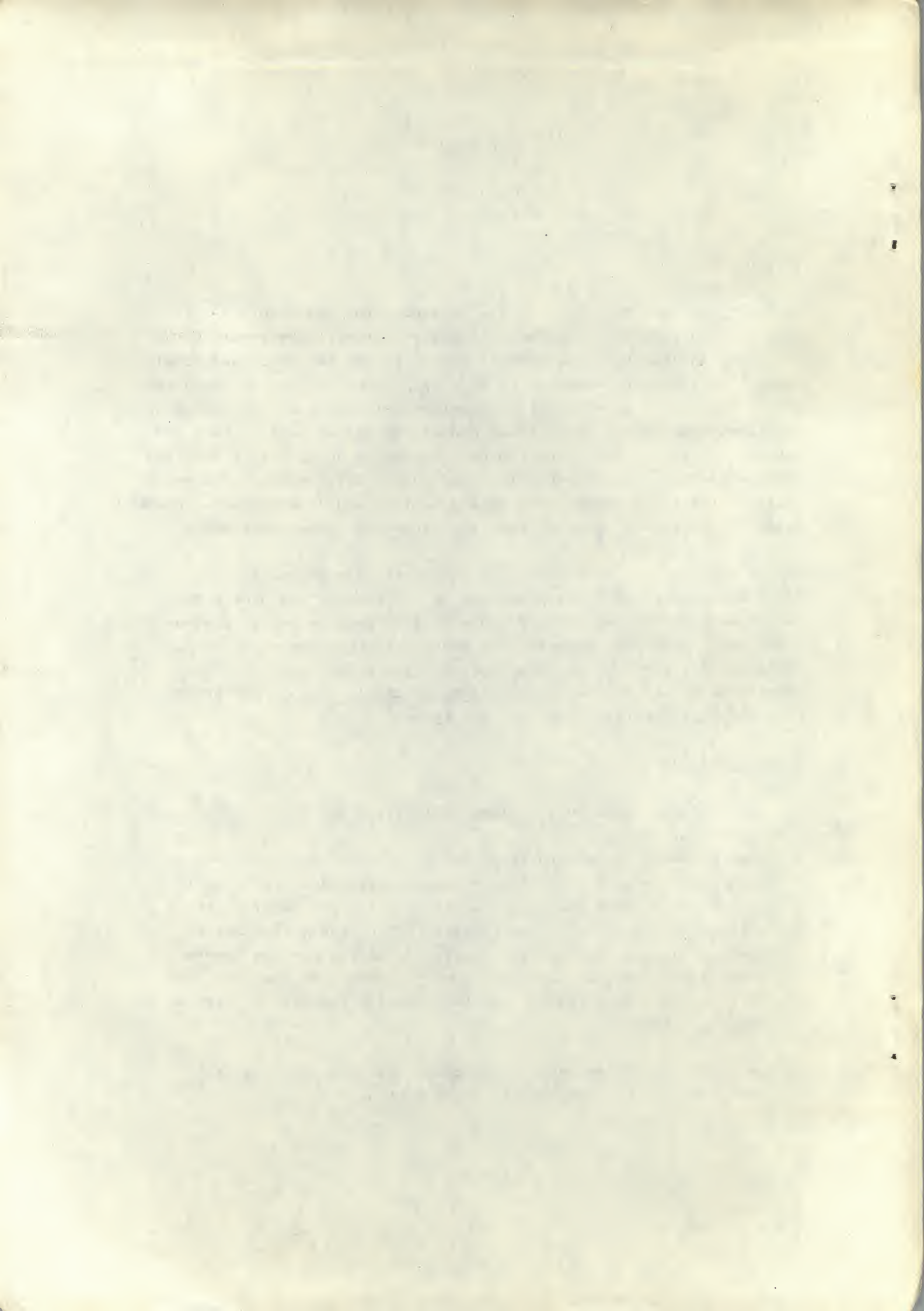
BASIC/RT provides the researcher with a powerful and complete software package for total programmable control of all LAB8/E peripheral devices: analog channels, Schmitt triggers, crystal clock and display scope. In addition, commands to specify parameters such as pulse rate and response time permit optimum experimental flexibility. Buffer allocation commands for the display and analog channels facilitate computer efficiency. Using the timing commands, a delay before sampling, a pause until a user response is typed on the Teletype, or a pause until an interrupt occurs, can easily be included in a program. Another feature, the user's command, permits customized system software.

All of the features provided by 8K BASIC are also implemented in BASIC/RT, permitting total programmable I/O control and user coded functions. The entire BASIC/RT command and function set is included in Appendix U of this manual. For additional information on BASIC fundamentals, refer to the EduSystem-10 System User's Guide; for details on 8K BASIC refer to the 8K BASIC User's Manual, DEC-08-SKXA-D. BASIC/RT does not run under the OS/8 system.

Loading Procedure

Use the following procedure to load BASIC/RT:

1. Load the RIM Loader into field 1. Either the high- or low-speed reader can be used. Set the data and instruction fields to 1 (use the EXTD ADDR LOAD key) before toggling in the first instruction. Note that after pressing DEP or EXAM, the LAB8/E always indicates the Program Counter location, not the Memory Address location, in the upper row of lights. To see what content has just been loaded, set the multiple register knob to MD (Memory Buffer).
2. Load the BIN Loader into instruction field 1. (Use the EXTD ADDR LOAD key to load in this field setting.)



Steps 1 and 2 can be replaced by loading the HELP Bootstrap.

3. Read in the BASIC/RT paper tape. Use data field 1 and instruction field 1. When loading via the BIN Loader, press START CLEAR then START CONT keys to initiate reading of the tape. When program is loaded, it prints the message READY.

1.0 SCOPE COMMANDS

The display scope on the LAB8/E can be programmed to plot points on its screen. The scope commands provide complete control for graph location and size, display time and number of points displayed.

1.1 PLOT

When plotting on the scope, which is rectangular, BASIC/RT uses these dimensions for its perimeter:

$$0 \leq X < 1.1$$

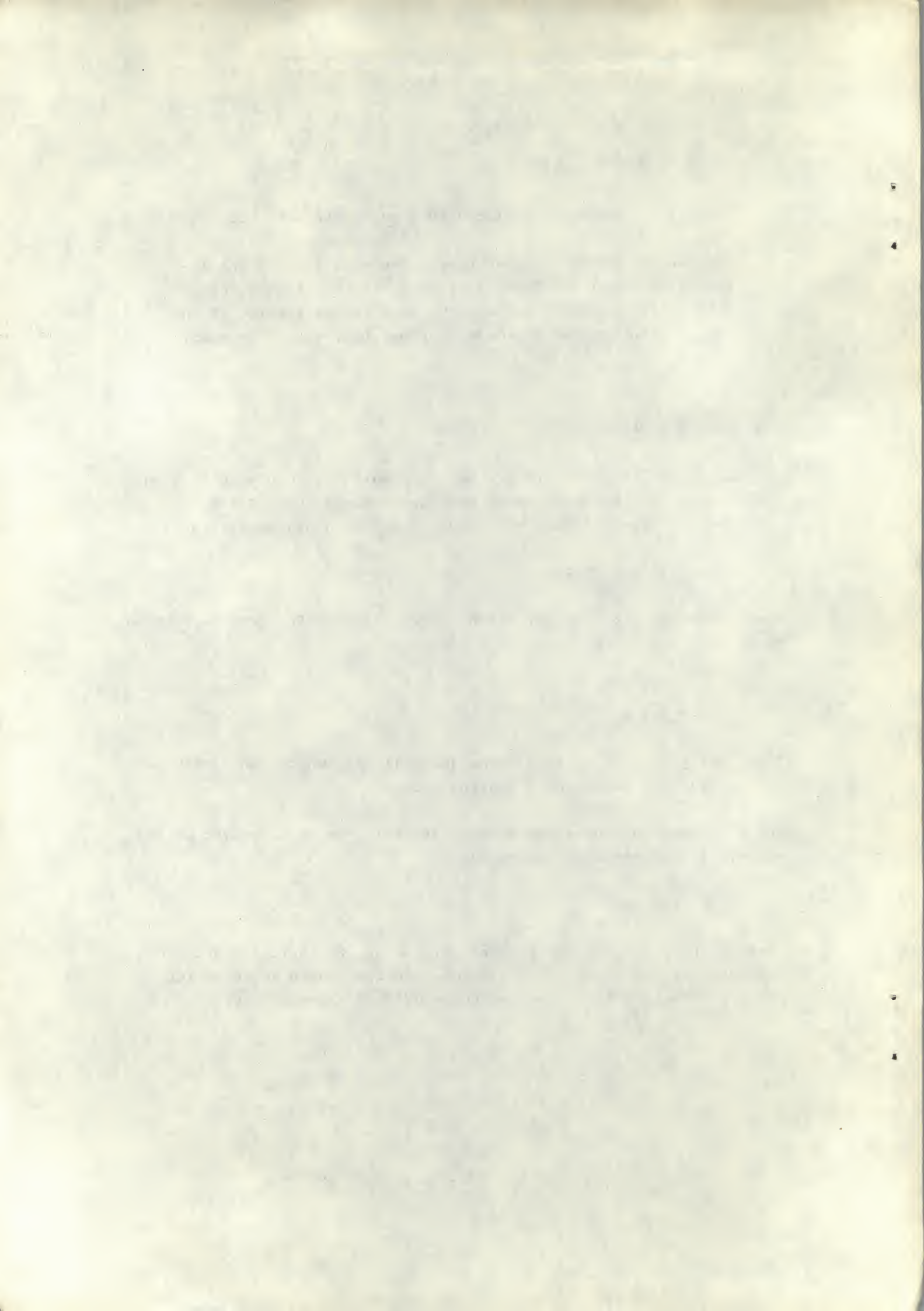
$$0 \leq Y < 1.0$$

Thus, any plot must be within the above limits, which can easily be accomplished by inserting a scaling factor.

The PLOT command causes the appropriate point to be displayed on the scope. It is issued in the format:

PLOT x,y

where x and y are any expressions which equal the actual X and Y coordinates of the point to be plotted. An acceptable sequence for plotting a straight line across the middle of the scope is:



```

5   B=.5
10  FOR A=0 TO 1.1 STEP .01
20  PLOT A,B
30  NEXT A

```

Remember that every X,Y set must be within the specified ranges. If it is not, that data set is simply not displayed.

BASIC/RT displays points on the scope when it is not doing any internal calculations. Data is displayed, for example, while waiting for input or output. If a calculation is required during a plotting sequence, the data is not displayed until all the calculations are completed. Thus, in the following example, plotting a decaying sine wave, the function is not displayed until all the points have been calculated.

```

200  FOR S=0 TO 1.10 STEP .006
210  PLOT S, SIN(S*35)*EXP(-S*2.5)/3+.5
220  NEXT S

```

1.2 DELAY

BASIC/RT provides a command that refreshes the scope after each calculation so that the progress of the graph can be seen. This command, DELAY, causes BASIC to display all x,y points calculated up to this statement. Thus, the decay of the sine wave above can be viewed after each point is calculated by adding to the above example the command

```

215  DELAY

```

The DELAY command provides the additional time for BASIC to display the point before continuing to the next statement.

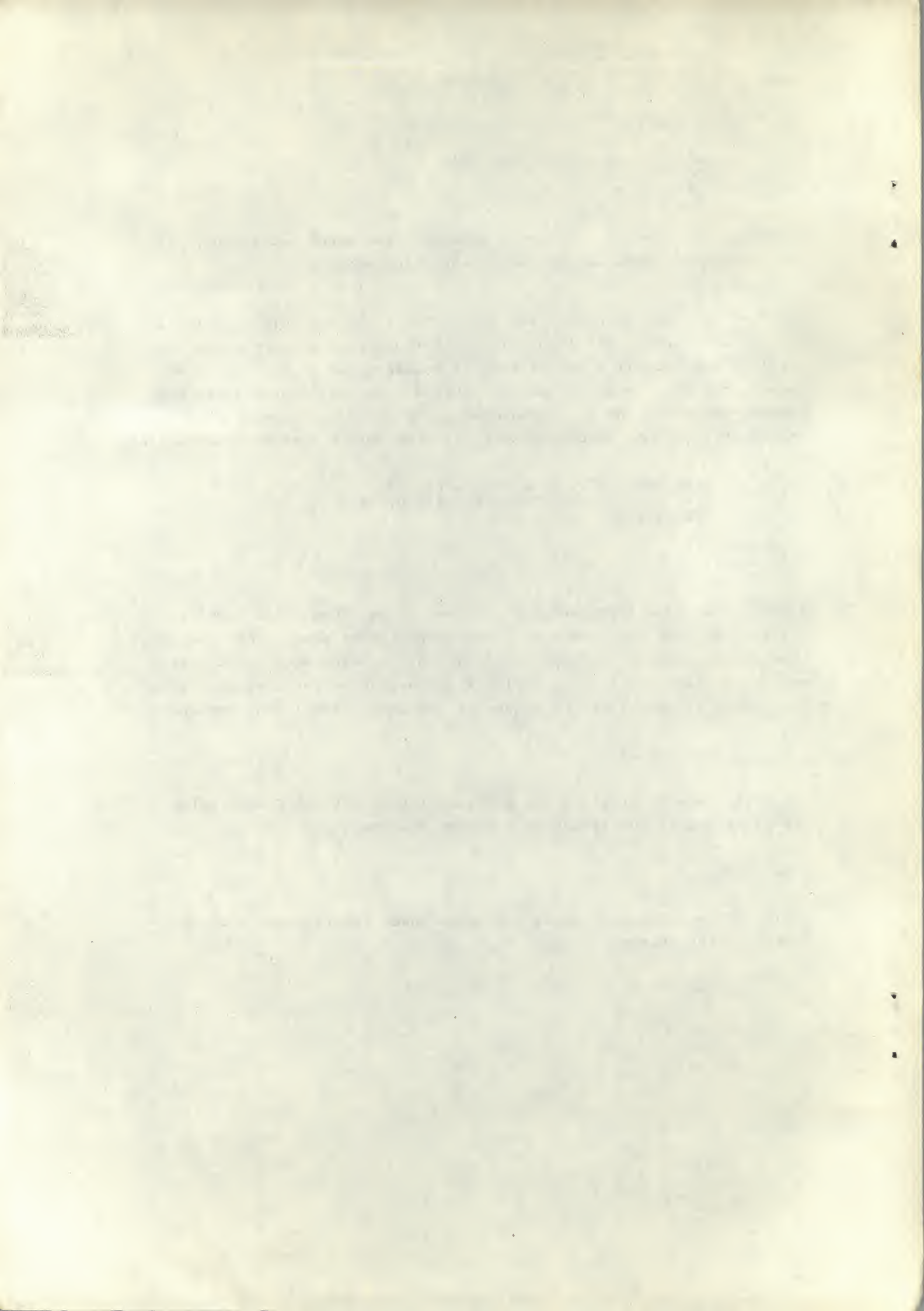
1.3 CLEAR

BASIC/RT also permits erasing the scope under program control. By inserting the statement:

```

CLEAR

```



all points currently displayed are removed from the screen. Thus, if more than one plot is required by a user program and it is not necessary for them to overlap, a CLEAR command between calculations erases the scope for the second plot.

In the next example, two compounded interest sums, \$400 at 7% and \$450 at 6.25% per annum compounded yearly for 30 years, are plotted. If the intercept point is to be noted, then line 120 can be omitted and at completion the two curves will be displayed together.

```
100 I=.07
102 P=400
104 T=30
110 GOSUB 500
120 CLEAR
130 I=.0625
134 P=450
140 GOSUB 500
150 STOP
500 FOR N=1 TO T
510 X=N/35
520 Y=(P*((I+1)^N)/40000)
530 PLOT X,Y
540 DELAY
550 NEXT N
560 RETURN
```

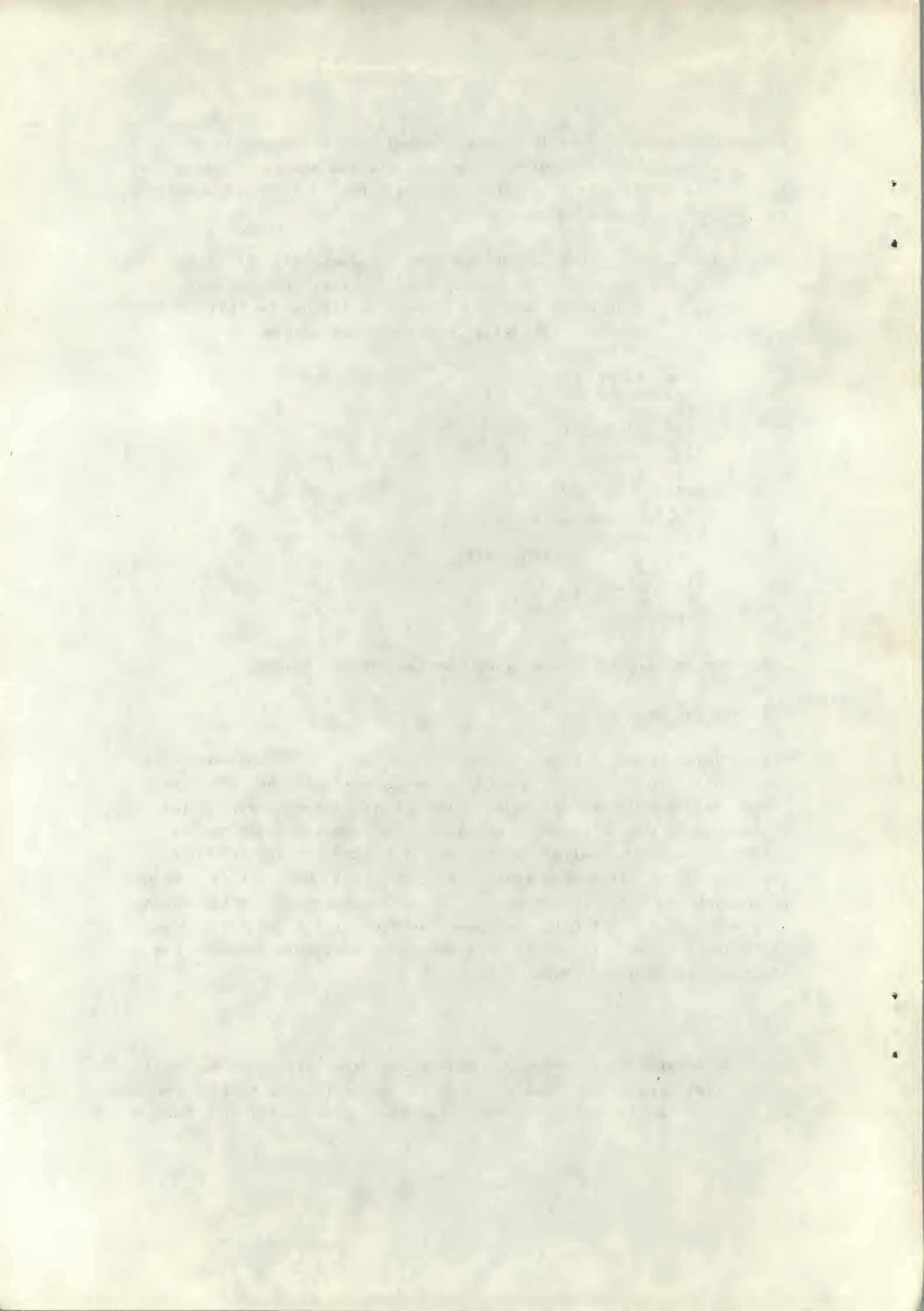
Lines 510 and 520 include scaling factors for the scope.

1.4 Display Buffer

The technique for plotting points employed by BASIC/RT includes creating a buffer in the user's area of core to store all the calculated points before they are displayed. (This buffer area is considered as a dimensioned variable; thus, executing the commands SCRATCH, RUN, and END removes the buffer from core. The CLEAR command does not delete the buffer, it merely erases its contents.) When a PLOT command is encountered, BASIC/RT checks to see if a display buffer has already been assigned and, if not, then space sufficient for about 500 points is allocated. If this amount of room is not available in core, the error message TOO BIG is printed.

1.5 USE

The area created by the PLOT command is approximately equal to a 333-dimensional array. In order to conserve space, if less than 500 points are to be plotted, or to plot more than 500 points, a buffer dimensioning



command is provided so that core can be allocated optimally. This command, USE x, is implemented as an array, as follows (x is always a variable):

```
20 DIM P(200)
30 USE P
```

Line 30 says : use P as a storage buffer for a future PLOT command; do not create an additional array at that time. Line 20 creates enough room for about 300 data points¹. If a user-assigned or BASIC/RT generated buffer is not large enough and overflows during execution the error message TOO BIG is printed.

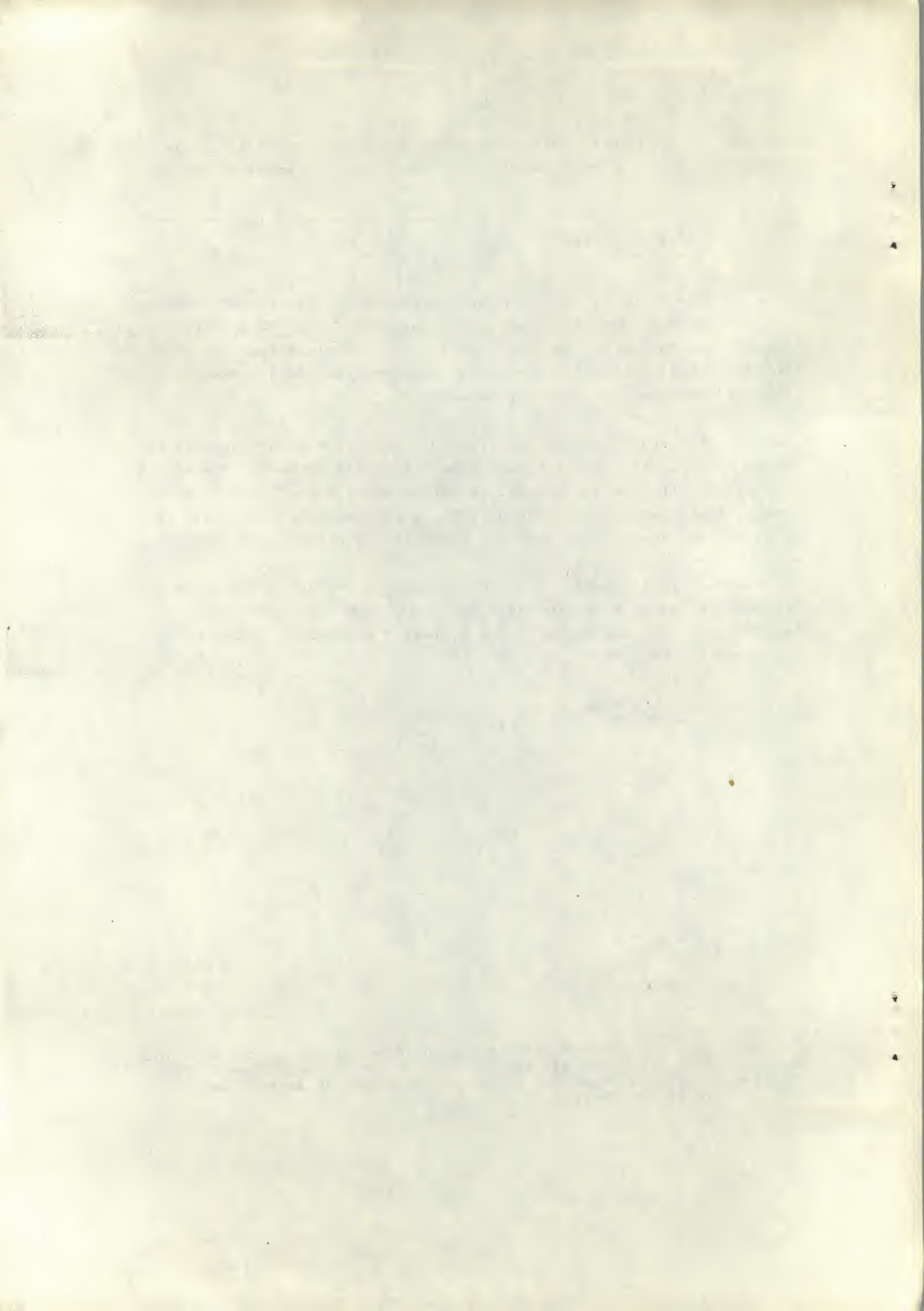
Only one USE statement is effective for any plot sequence and, if it is to be used, it must be issued before the PLOT command. The variable associated with the USE command is active until one of the buffer removing statements (RUN, SCRATCH, END) is encountered, but it can be used for another purpose when not currently required for displaying.

In the example in Section 1.3, the maximum number of points to be plotted is 30, so a considerable amount of core can be conserved by specifying a smaller buffer. The following code can be added to optimize the program.

```
10 DIM M(40)
20 USE M
```

¹To determine the minimum array size for point displays, consider that each point requires two values (X and Y) and that every three values require one word of memory. To display P points an array dimensioned for W words is necessary:

$$W = .67(P)$$



CHAPTER 2

CLOCK COMMANDS

2.0 GENERAL DESCRIPTION

The hardware clock provided in the LAB8/E, like all the other devices, can be handled under program control to maximize its performance, by specifying parameters such as pulse rate and initial pulse time.

2.1 SET RATE

To set the clock to interrupt at a specific rate, use the command

SET RATE mode,time

where mode¹ is the desired clock speed (0-7) and time is the number of clock "ticks" between interrupts, up to 4096 counts. The appropriate mode value is derived from the next chart².

mode	rate
0	Stop
1	external input
2	10 ⁻² seconds
3	10 ⁻³ seconds
4	10 ⁻⁴ seconds
5	10 ⁻⁵ seconds
6	10 ⁻⁶ seconds
7	Stop

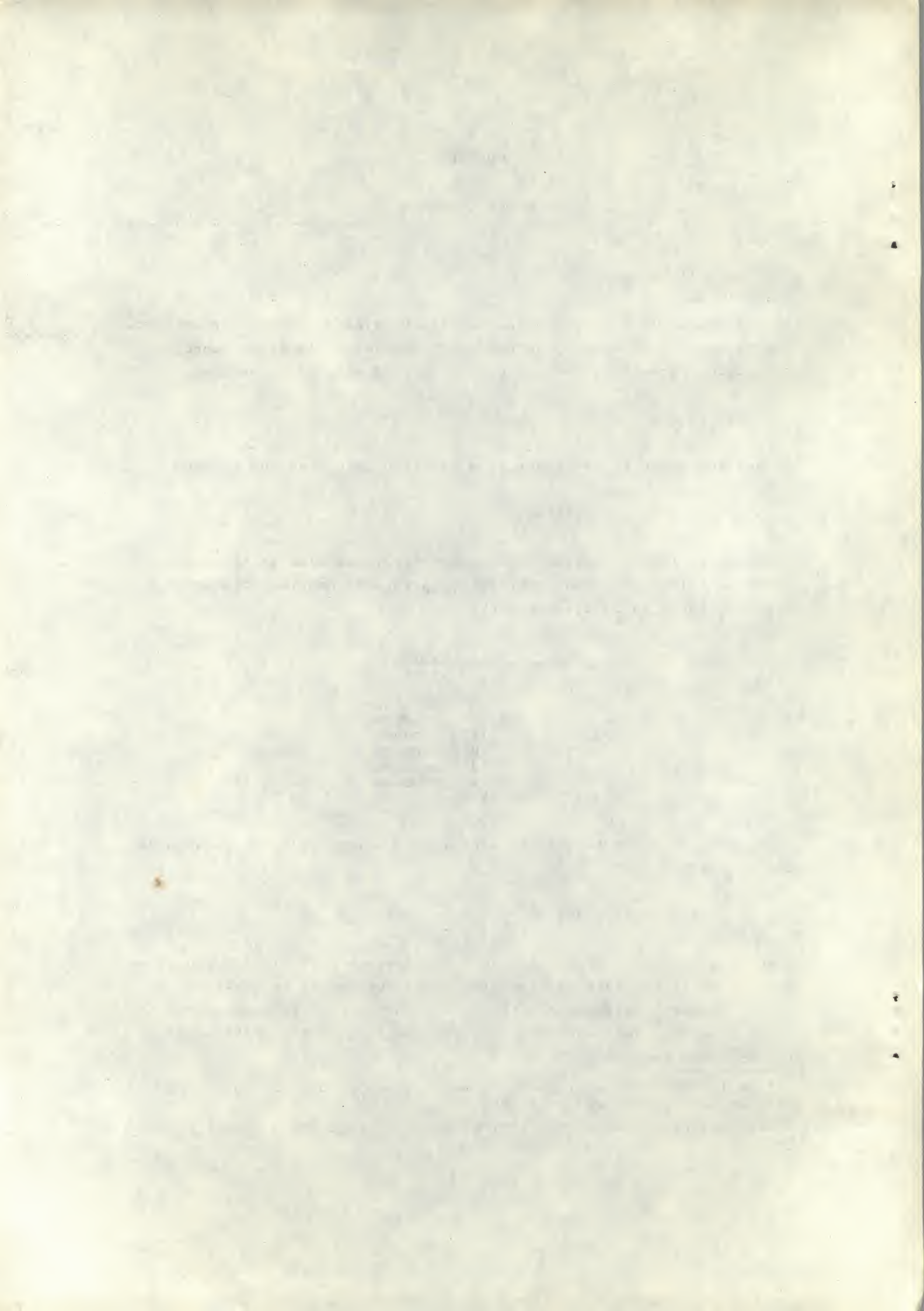
Thus, for the clock to interrupt at 1 second intervals, an acceptable command is

SET RATE 3,1000

which causes the clock to wait 1000 one msec. ticks. If the specified clock rate is too fast, so that the interrupt cannot be serviced in time, the error message RATE ERROR AT (line number) is printed and the clock stopped. The line number printed is that of the statement currently being executed.

¹If an illegal mode is requested, BASIC tries to use the value: -3 will be taken as 3, .03 as 0, etc.

²Refer to DK8-EP Clock Description in the PDP-8/e Small Computer Handbook.



Note that in the 100-200 μ sec elapsed total time range BASIC/RT is servicing the interrupt correctly, but is not executing any BASIC/RT commands because of the high rate; in this case, the processing has been suspended. Type CTRL/C, which will cause the RATE ERROR message to be printed, to restart.

2.2 SET CLOCK

BASIC/RT provides another command for setting the clock rate for the Schmitt triggers. Its format is the same as that for the SET RATE command, namely

SET CLOCK mode,time

except that mode is a 12-bit decimal number which will be used to load the clock enable register¹, thus permitting the user to enable any function he chooses. Refer to Table 1 to determine the appropriate value. The time parameter is specified in the same manner as with the SET RATE command².

Any time either of the SET statements is encountered, the time counter is zeroed. Then, any time a clock interrupt occurs, this counter is incremented. Up to approximately 16,000,000 counts can occur before this counter resets itself to zero.

2.3 TIM

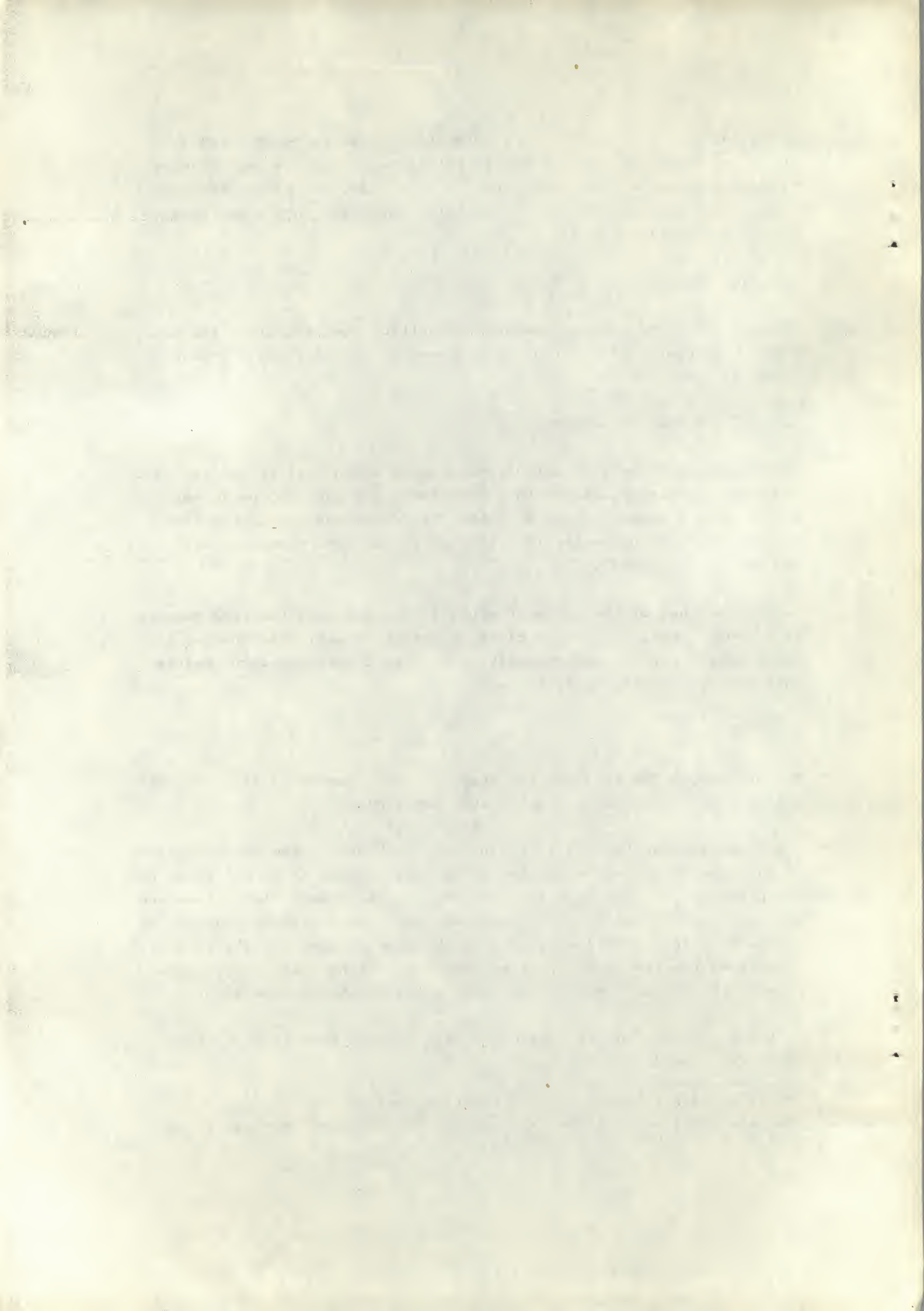
At any time in the program, the current count (number of elapsed interrupts) can be determined via the function TIM(n).

This function can be used in conjunction with any of the BASIC commands so that the value can be printed or the next action to be performed by the program can be dependent on the count. The format of the function call is TIM(n) where n is any argument (the argument is not checked by BASIC/RT). In the following program the elapsed time for the plot determines the next action; print the count and halt or, for 50 data points, print the sines and the terminating count and then stop.

In the first run, the time elapsed. By changing line 10, the sine table was generated.

¹BASIC/RT always forces the interrupt request bit on.

²The 2's complement of the value of time is performed so that it is in a form suitable for the clock.



10	A=10	RUN
20	SET RATE 2,20	10
30	FOR M=0 TO 1 STEP .01	READY.
40	PLOT M,M+2	
50	DELAY	
60	NEXT M	10 A=12
70	IF A>TIM(0) THEN 200	RUN
80	PRINT TIM(23)	0
90	STOP	.01999867
200	FOR Z=0 TO 1 STEP .02	.03998933
210	PRINT SIN(Z)	.
220	NEXT Z	.
230	PRINT TIM(C)	.
240	END	

2.4 WAITC

Another application of the clock is to halt program execution until a clock interrupt occurs. The WAITC command performs this function, thereby permitting BASIC/RT to display on the scope while waiting for the interrupt to signal continuation of program execution.

2.5 CLS and CLC

There are two other functions that are used with the Schmitt triggers. The first, CLS(n), obtains the status bits of the clock after the previous interrupt (those obtained from the PDP-8 CLSA (Clock Status to AC) instruction). This value can be tested, for example, to determine how the interrupt occurred. The other, CLC(n), performs a CLCA (Counter to buffer to AC) and returns the value as a floating point number. Refer to PDP-8/e Small Computer Handbook.

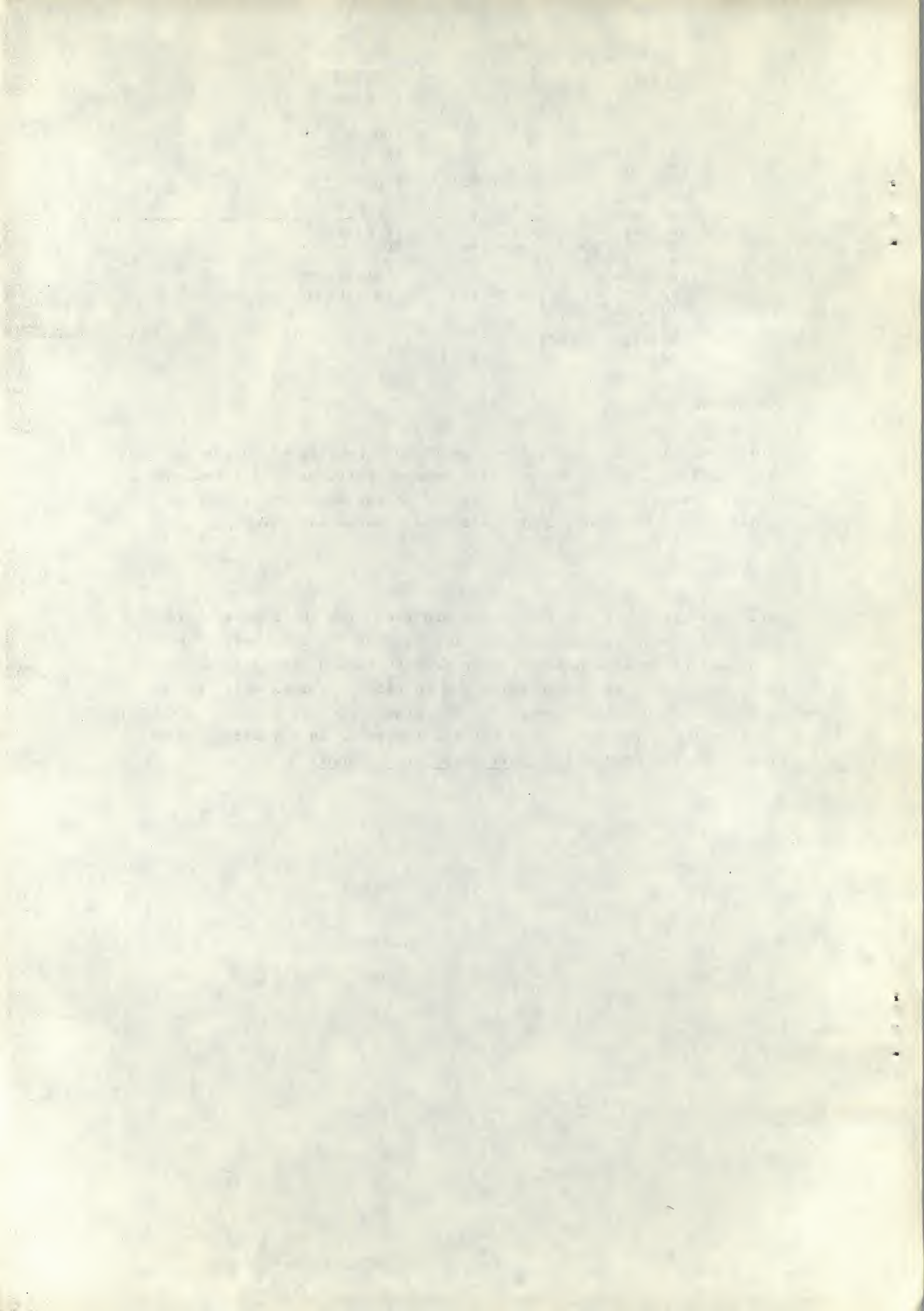
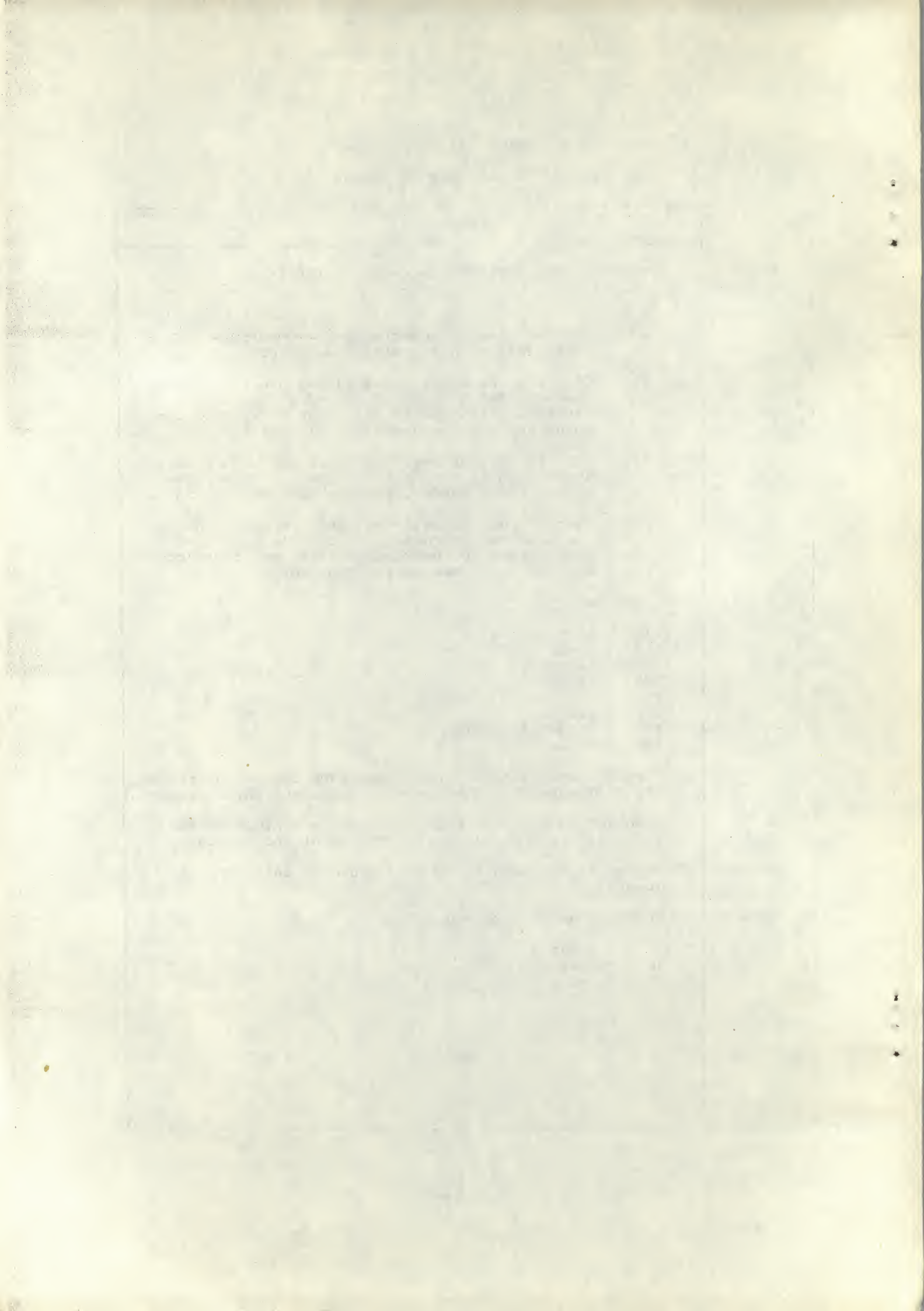


Table 7-1
Clock Enable Register Functions

AC Bit	Function
0	Enables clock overflow to cause an interrupt.
1 & 2	<p>Mode</p> <p>00 Counter runs at selected rate. Overflow occurs every 4096 counts. Flag remains set.</p> <p>01 Counter runs at selected rate. Overflow causes Clock Buffer to be transferred to the Clock Counter, which continues to run. Overflow remains set until cleared with IOT 6135.</p> <p>10 Counter runs at selected rate. When an enabled event occurs, the Clock Counter is transferred to the Clock Buffer, and the Counter continues.</p> <p>11 Counter runs at selected rate. When an enabled input occurs on channel 3, the Clock Counter is transferred to the Clock Buffer, and the Clock Counter continues to run from zero.</p>
3, 4 & 5	<p>Count Rate</p> <p>111 Stop</p> <p>110 1 MHz</p> <p>101 100 KHz</p> <p>100 10 KHz</p> <p>011 1 KHz</p> <p>010 100 KHz</p> <p>001 External input</p> <p>000 Stop</p>
6	Overflow starts ADC. (When the Clock Counter overflows, the analog-to-digital converter, type AD8-EA, is started.)
7	When set to a 1, the Crystal Clock is inhibited from generating clock pulses that increment the counter.
8	Events in Channels 1, 2, or 3 cause an interrupt request.
9, 10 & 11	<p>Enable Events 1, 2, and 3</p> <p>9 - Event 3</p> <p>10 - Event 2</p> <p>11 - Event 1</p>



CHAPTER 3

A/D COMMANDS

3.1 ADC

Any A/D channel can be sampled at any time by using the single function `ADC(n)`, where `n` is the channel number, 0 to 15, to be sampled in a direct or indirect statement. The ADC function performs an immediate conversion; the clock, however, can be incorporated so that sampling occurs at an established clock rate. In the next program, BASIC/RT waits for a clock tick and then prints the value of the clock using the `TIM` function, and of A/D channels 3 and 4 for fifty samples.

```
300 SET RATE 2,60
310 FOR P=1 TO 50
320 WAITC
321 A1=ADC(3)
322 A2=ADC(4)
323 T1=TIM(0)
330 PRINT T1,A1,A2
340 NEXT P
350 STOP
```

3.2 REAL TIME

The ADC function is restricted to non-time critical work because the finite amount of time elapsed between clock ticks may not be sufficient to perform the tasks requested between ticks (for example, printing three values above). Also, more than one channel cannot be sampled in the same time quanta (for example, sampling channels 3 and 4 above).

For time critical operations, the `REAL TIME` command should be used because it provides a buffer to hold the sampled value prior to processing. Its format is:

```
REAL TIME v,c1,n1,n2
```

where `v` is a subscripted variable to be used as the data buffer. The variable is assigned in a manner analogous to the `USE` command for the scope, namely, as a dimensioned array. Because only one value is to be taken per sample, three samples are stored per buffer word. Thus, a dimension of 100 can store 300 data items. The array cannot be dimensioned larger than approximately 750, or a maximum of approximately 2,200 points. The parameter `C1` is the first channel to be sampled,

The first of these is the fact that the
the second is the fact that the
the third is the fact that the

The fourth is the fact that the
the fifth is the fact that the
the sixth is the fact that the

The seventh is the fact that the
the eighth is the fact that the
the ninth is the fact that the

The tenth is the fact that the
the eleventh is the fact that the
the twelfth is the fact that the

The thirteenth is the fact that the
the fourteenth is the fact that the
the fifteenth is the fact that the

n1 is the number of consecutive channels to be sampled, and n2 is the number of clock ticks for which to sample. To prepare to sample channels 1 and 2 once every millisecond for 150 milliseconds, suitable code is:

```
SET RATE 4,10
DIM G(100)
REAL TIME G, 1,2,150
```

Operation of the REAL TIME command is independent of the BASIC/RT statement processing speed; as long as there is sufficient buffer space, the REAL TIME statement will work.

3.3 ACCEPT and REJECT

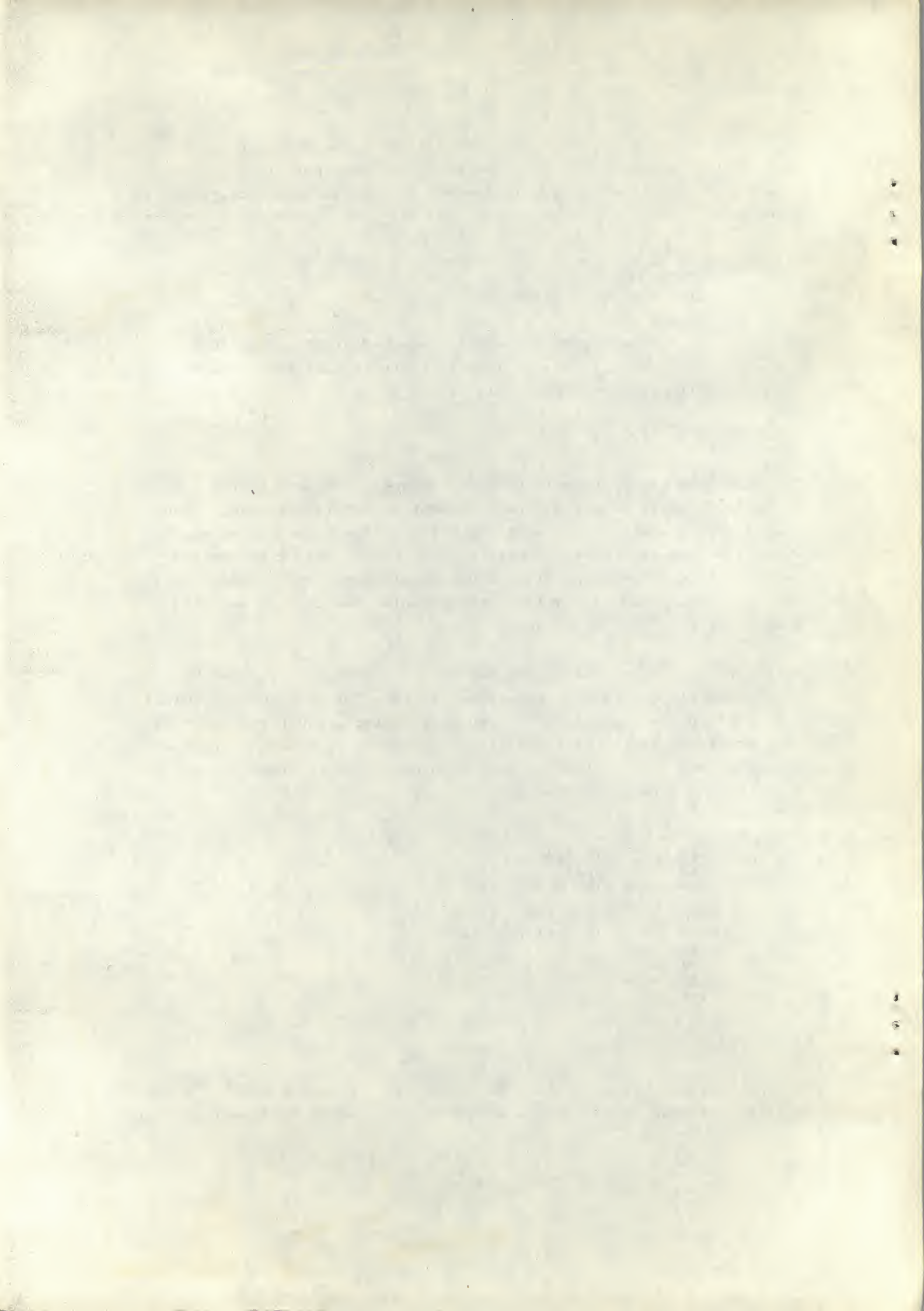
The REAL TIME statement only creates the specified data buffer. To actually initiate sampling, the statement ACCEPT is required. Then sampling will start at the next clock tick. There must be an active REAL TIME statement or the ACCEPT is ignored. A REAL TIME command becomes inactive when the clock count equals zero. To suspend sampling use the command REJECT. This command is also useful for executing subsequent REAL TIME statements.

In the next example, the ACCEPT and REJECT statements are used to be sure sampling occurs at the specified rate for only the designated number of counts. Statement 500 stops the clock, 530 prepares it so that the first sample is taken after processing statement 540. The REJECT at 560 assures that after 100 counts no extra samples are taken at the rate of 10 mseconds.

```
500 SET RATE 7,0
510 DIM G(100)
520 REAL TIME G,1,2,150
530 ACCEPT
540 SET RATE 2,10
550 IF TIM(4)<100 THEN 550
560 REJECT
570 SET RATE 3,10
580 ACCEPT
590 END
```

3.4 Buffer Capacity

In the last REAL TIME example, the buffer, G, is subscripted to accommodate $3 \times 100 = 300$ items. The program, however, could have been required



to sample a greater total number of points. If the 520 was REAL TIME G, 1,6,150 then a total of $6 \times 150 = 900$ items must be accommodated. To handle this variation without having to allocate large amounts of core as buffer space, BASIC/RT uses a ring buffer so that data can be removed from the buffer before it is overwritten by a new, incoming item. To meet this condition, the sampling can be:

1. At a slow rate for a long time (consistent time sampling) or
2. At a fast rate for a short time.

If a fast rate is required for a longer period, a large buffer can be created, provided this space is not required by BASIC/RT for variables, etc. If the buffer becomes completely full such that data is being overwritten, the message A-D FULL is printed. The maximum throughput rate is about 4 KHz or 250 μ s.; the steady state long term rate is about 50 points/second.

3.5 ADB

To retrieve data collected by REAL TIME and ACCEPT sequences that are placed in a buffer, the ADB(n) function is required. Data is withdrawn from the buffer in the same order in which it was entered. Thus, if four A/D channels, 1-4, are being sampled, the order of the data in the buffer is

$$1_i 2_i 3_i 4_i \quad 1_{i+1} 2_{i+1} 3_{i+1} 4_{i+1} \quad 1_{i+2} \dots$$

The argument in the ADB function is ignored. The items will be removed consecutively. If there is no REAL TIME or ACCEPT statement or if there is no data remaining in the buffer (because the number of clock ticks, n2, in the REAL TIME statement had expired), the error message NO A-D and a line number are printed.

This example is an expansion of the one in Section 3.3 to incorporate the ADB function. Lines 590 through 610 have been added so that the sampled values can be printed.


```

500 SET RATE 7,0
510 DIM G(100)
520 REAL TIME G,1,2,150
530 ACCEPT
540 SET RATE 2,10
550 IF TIM(4)<100 THEN 550
560 REJECT
570 SET RATE 3,10
580 ACCEPT
590 FOR A=1 TO 150
600 PRINT ADB(1),ADB(2)
610 NEXT A
630 END

```

The user must keep track of the items in the buffer (number requested and number removed) and their ordering by channel number. In the above example there will be 75 lines printed, each with two data values in the format:

```

1i      2i
1i+1    2i+1
1i+2

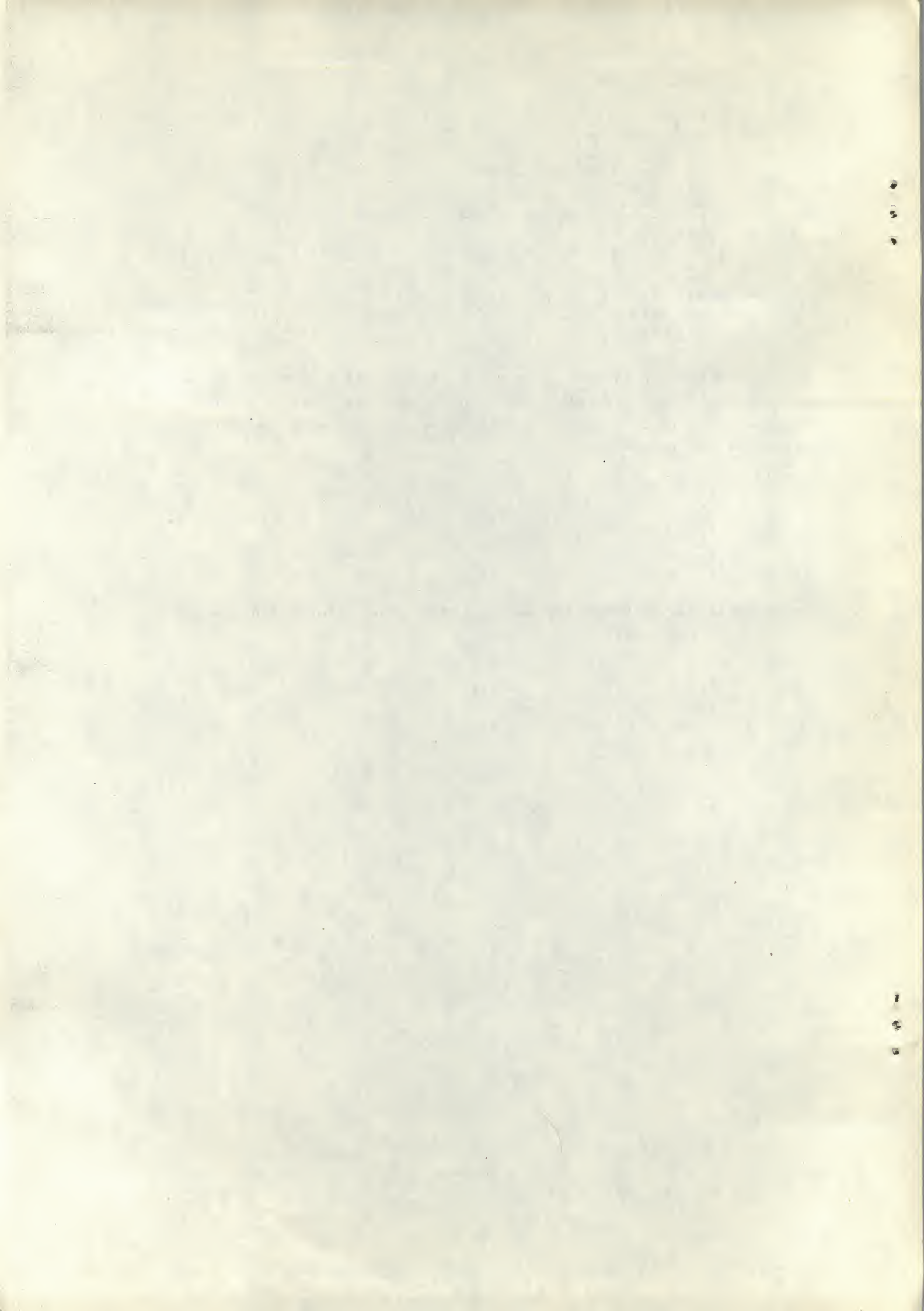
```

If line 140 was PRINT ADB(0),ADB(6),ADB(2), the information printed would be in the order:

```

1i      2i      1i+1
2i+1    1i+2    2i+2
1i+3

```



CHAPTER 4

TEST AND PAUSE COMMANDS

4.1 TST

For automated environments requiring a response typed on the Teletype to determine the subsequent action, a testing function, TST(n), is available which checks the Teletype to see if a character has been typed. If it has, a non-zero is returned; if it has not, a zero is returned. The TST function is particularly handy in conjunction with the TIM function for specifying a maximum response time. The TST function must be reset by doing a GETC function for the TST statement to be used more than once in a program.

4.2 WAIT

If a pause time is required by a program, the WAIT command can be included in the program. BASIC/RT processing will be halted until any interrupt occurs. Note that a clock interrupt is sufficient to reactivate BASIC/RT.

CHAPTER 5

User Commands and Functions

5.1 User Command (UCOM)

The USER COMMAND is a feature in BASIC/RT which allows the user to insert his own code and to effectively communicate with BASIC/RT. This command differs from the UUF function in that it is more sophisticated and hence more complicated in its operation. To briefly explain the differences: UUF is a function, such as SIN and COS, while UCOM is a command. A function takes an expression in parentheses as its argument and returns a value to be used later. For example, Y=INT(4.5) takes 4.5 and returns 4.0. A command is a statement which may take arguments (such as READ or INPUT), or may not have any (such as CLEAR or END), or they may be optional (as with PRINT). In a function such as UUF, BASIC/RT has already evaluated the expression and left the results in the FAC before calling the function so the function has no control over what is in it. With a command, it is up to the user code to scan and determine what is wanted. To successfully use this command, a knowledge of PDP-8 code and the floating point package is essential. It is suggested that the coding for commands such as CLEAR, PLOT, USE, (and any others) be studied in order to understand the nature of the commands. Some brief examples are illustrated next to show how to insert some simple commands.

In this example, a UCOM is created such that whenever it is issued, it executes a 6123 (some random IOT) with the AC all 7's. The first thing to do is allocate space. There is room in upper and lower core. In this case, let us overlay the EXP function with the UCOM. Put a pointer in the execute table to the location of the command. The table entry will be about location 306 (refer to a listing for the exact location). Now, whenever a UCOM is given, control will transfer to where this location points. The code will then look like:

*EXP	
CLA CMA	/Set the AC to all 7's
6123	/Execute the IOT
CLA	/Clear the AC
JMP I (DEVCOM)	/and continue with BASIC

The routine DEVCOM checks to see that the next item on the line is a CR (carriage RETURN). If it is not, then an error message will be given. This is to prevent illegal user code from getting through: UCOMING HOME FOR DINNER? The previous sentence would call the UCOM processor, but since the next item is not a carriage RETURN, an error message would be given.

There are two more routines that can be used: DEVCON and SKIPIT. DEVCON does the same thing as DEVCOM except that it checks the present item for CR rather than the next item. SKIPIT skips past the rest of the line until it finds a CR and never gives an error message.

To modify the previous example to take two expressions and multiply them together and then send out the low 12 order bits to the device via a 6123, the command would look like:

UCOM A,B

where A and B are any expressions. The coding is:

*EXP		
JMS	GETWD	/skip past the UCOM element.
JMS I	(MEVAL	/evaluate next element of line.
FENTER		
FSTA	TEMP	/save away
FEXIT		
JMS I	(MEVAL	/evaluate next element of line.
FENTER		/enter the FPP package
FMU	TEMP	/multiply A*B.
FEXIT		/exit now.
JMS I	(FIX	/fix the AC now.
CLA		/clear the AC.
TAD	AC3	/pick up low word.
6123		/send to the device.
CLA		
JMP I	(DEVCON	/and exit.

Notice that the exit is via DEVCON rather than DEVCOM. This is because MEVAL has evaluated the expression until it found a delimiter. This must be a comma in the first place and a CR in the second place. If it is not, then DEVCON gives an error message. As stated previously, the user should look at PLOT and USE and perhaps a few other commands to fully appreciate the UCOM command.

It is possible to put the UCOM code in field 1 with only some minor changes. Since some commands are dispatched again in field 1, place

in 306 a pointer to 7743. (7743 is the upper core dispatcher.) Then, in field 1 at 744, place the address of the UCOM command code. On UCOM, BASIC/RT will now go to this location in field 1. Cross-field communication is done through a variety of functions, all of which are mentioned in the internal documentation. The novice user is advised against putting instructions in field 1 until he has successfully run functions and commands from field 0. Note that functions can also reside in field 1. The user should examine ACCEPT, TIM and possibly REAL TIME to get a better understanding of cross-field communications.

It is possible to allocate more space in field 1 for functions/commands by changing the value of LIMIT. LIMIT is the first location of upper core used by BASIC/RT. By changing PLIMIT at 2562 to, say, 2000, about 1000₈ locations are gained for user functions and commands. Note that this space is lost from the user program and variable area, but very often this is worth the sacrifice.

5.2 User Coded Function

As in 8K BASIC, the user can include his own function in BASIC/RT, calling it as UUF(n), to perform a special task. The UUF function is coded in the same manner as described in the 8K BASIC User's Manual, except that the suggested locations for the code are 0064 to 0177 in field 1. Other smaller areas are available in field 0 and many more can be created by deleting some of the mathematical functions.

CHAPTER 6

ERROR MESSAGES

The computer checks all commands before executing them. If for some reason it cannot execute the command, it reports the fact by printing one of the messages in Table 7-2 and the number of the line in which the error was found.

Table 7-2
BASIC/RT Error Messages

Message	Explanation
SYNTAX ERROR	The command does not correspond to the language syntax. Common examples of syntax errors are misspelled commands, unmatched parentheses, and other typographical errors.
FUNCTION ERROR	A function was used which was deleted when the system was loaded and hence is not available. A DEF statement will produce this error if the DEF capacity has been deleted.
TOO-BIG ERROR	The combination of program and variables exceeds the capacity of the computer. Reducing one or the other may help. If the program has undergone extensive revision, try punching it out, typing SCRATCH, and reloading.
SUBSCRIPT ERROR	A subscript has been used which is outside the bounds defined in the DIM statement.
LINENO ERROR	A GOTO, GOSUB, or IF references a nonexistent line.
FOR ERROR	FOR loops are nested too deeply.
NEXT ERROR	FOR and NEXT statements are not properly paired.
GOSUB ERROR	Subroutines are nested too deeply.
RETURN ERROR	GOSUB and RETURN statements are not properly paired.
DATA ERROR	There are no more items in the data line.
ARGUMENT ERROR	A function has been given an illegal argument, for example, SQR(-1).
RATE ERROR	Clock rate was too fast to service interrupts.
A-D FULL	No more room in the data buffer for remaining items.
NO A-D	No input coming from specified analog channel.

